
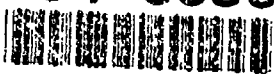


REPORT DOCUMENTATION PAGE

OMB No 0704 0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (leave blank)	2. REPORT DATE August 3, 1994	3. REPORT TYPE AND DATES COVERED Technical Report 05/01/94-08/01/94
4. TITLE AND SUBTITLE COLOR IMAGE PROCESSING USING CELLULAR NEURAL NETWORKS		5. FUNDING NUMBERS G N00014-94-1-0516
6. AUTHOR(S) Jose Pineda de Gyvez AD-A286 455 		8. PERFORMING ORGANIZATION REPORT NUMBER 94-35961 
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Texas Engineering Experiment Station Texas A&M University		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research Code 215: JWK Ballston Tower One 800 North Quincy Street Arlington, Virginia 22217-5660		
11. SUPPLEMENTARY NOTES Results published in Electronics Letters		
12a. DISTRIBUTION/AVAILABILITY STATEMENT A: Approved for public release: distribution unlimited		12b. DISTRIBUTION CODE DTIC SELECTED NOV 23 1994 F
13. ABSTRACT (Maximum 200 words) This report addresses the functional behavior of Cellular Neural Networks (CNN). The impact of variable convergence times on the proper operation of the network is discussed. A test method is presented to determine the functionality of the network. The function fault models assume that the cells are unable to switch between limiting states. The proposed method attains 100% stuck-at fault coverage without any extra hardware for its implementation. Moreover, the required number of test vectors is constant and independent of the array size which makes it suitable for practical implementations. The report discusses the new fault model, presents the algorithmic procedures and shows simulated testing results.		
14. SUBJECT TERMS Cellular Neural Networks, Testing		15. NUMBER OF PAGES 11
		16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified
20. LIMITATION OF ABSTRACT		

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

94 1120 033

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to stay within the lines to meet optical scanning requirements.

Block 1. Agency Use Only (Leave blank)

Block 2. Report Date Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers To include contract and grant numbers, may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C - Contract	PR - Project
G - Grant	TA - Task
PE - Program Element	WU - Work Unit
	Accession No

Block 6. Author(s) Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es) Self-explanatory.

Block 8. Performing Organization Report Number Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es) Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number (if known)

Block 11. Supplementary Notes Enter information not included elsewhere such as: Prepared in cooperation with; Trans. of; To be published in. When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD - See DoD 5230.24, "Distribution Statements on Technical Documents."

DOE - See authorities.

NASA - See Handbook NHB 2200.2

NTIS - Leave blank.

Block 12b. Distribution Code

DOD - Leave blank.

DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.

NASA - Leave blank.

NTIS - Leave blank.

Block 13. Abstract Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report.

Block 14. Subject Terms Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages Enter the total number of pages.

Block 16. Price Code Enter appropriate price code (NTIS only).

Blocks 17 - 19. Security Classifications Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED) if form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

**Hardware Implementation of a
Desktop Supercomputer for
High Performance Image Processing**

ONR Grant Number N00014-94-1-0516

**Technical Report
Aug/01/94 - Nov/01/94**

Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

**BEHAVIORAL TESTING OF
CELLULAR NEURAL NETWORKS**



Dr. Jose Pineda de Gyvez

Texas A&M University

Microelectronics Group

Department of Electrical Engineering
College Station, TX, 77843

Phone: (409) 8457477

FAX: (409) 8457161

Email: gyvez@pineda.tamu.edu

TABLE OF CONTENTS

1. Introduction	3
2. Background Theory	3
3. Convergence Variance	4
4. Convergence Test Method	5
5. Simulated Convergence Test	6
6. Fault Models	6
7. Functional Test Methods	6
8. Simulated Functional Test Results	7
9. Conclusion	9
References	9

1. Introduction

The converged final state of a Cellular Neural Network relies on the interaction between cells as determined by the templates. Some applications depend on the output value of cells to determine the final state of the Network. This paper will look at the effect of variances in the convergence rate of cells and its impact on the final state of the Network.

The testing of CNNs has been scarcely addressed. The only existing approach has limited applications such as only orthogonal interaction with neighboring cells is tested and it requires additional hardware to implement [1]. The test described in this article overcomes the previous limitations and achieves 100% fault detection. Using the concept of C-Testability, it is possible to determine the functionality of a processing array by applying a constant number of predetermined vectors independent of the array size and then comparing the actual output values to the predicted output values [2]. Under C-Testability, the input is propagated through the network to arrive at a final output state. If the actual final state is the one predicted by the given input vector, then the network is determined to be operating properly. However, if a given cell is faulty, its faulty state value will also be propagated and the fault will appear at the output.

2. Background

A CNN is an analog cellular nonlinear dynamic processor array. The basic circuit unit is called the *cell* [3,4]. It contains linear and nonlinear circuit elements. Any cell, $C(i,j)$, is connected only to its neighbor cells, i.e. adjacent cells interact directly with each other, see Fig. 1. This intuitive concept is called *neighborhood* and is denoted as $N(i,j)$.

Cells not in the immediate neighborhood have an indirect effect because of the propagation effects of the dynamics of the network. Each cell has state x , input u , and output y . The state of each cell is bounded for all time $t > 0$ and, after the transient has settled down, a cellular neural network always approaches one of its stable points. The dynamics of a cellular neural network have both output feedback (A) and input control (B) mechanisms. The first order nonlinear differential equation defining the dynamics of a cellular neural network is shown by (1).

$$C \frac{dx_{ij}(t)}{dt} = -\frac{1}{R} x_{ij}(t) + \sum_{C(k,l) \in N(i,j)} A(i,j;k,l) y_{kl}(t) + \sum_{C(k,l) \in N(i,j)} B(i,j;k,l) u_{kl} \quad (1)$$

$$y_{ij}(t) = \frac{1}{2} (|x_{ij}(t)| + 1 - |x_{ij}(t)| - 1)$$

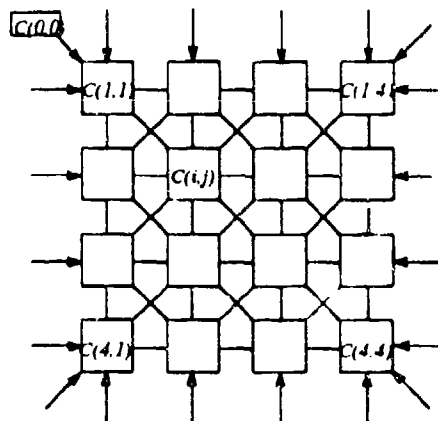


Fig. 1 Cellular Neural Network 4 x 4 processing array with border cell inputs

A set of inputs is necessary to simulate interaction with imaginary cells outside the processing array to insure that the cells on the perimeter of the processing array achieve proper convergence. These imaginary cells are called border cells and form a ring around the processing array. A border cell, such as border cell $C(0,0)$ of Fig. 1, outputs a constant voltage to mimic the output voltage v_y that a properly converged cell would produce as well as a voltage v_u to represent the input image voltage that border cell $C(0,0)$ would receive if it were a functioning cell. Therefore, a cell on the perimeter of the processing array uses the image voltage and dynamic output voltage of neighboring cells as well as the static output and image voltages of the border cells to arrive at the proper final state. The border cells are treated as members of the array for initialization purposes and template implementation, but are not considered in the final state analysis.

3. Convergence Variance

Cell convergence is achieved when $C \frac{dx_y(t)}{dt} = 0$. The rate of convergence is determined by many factors. The amount of current that is flowing into the cell, as governed by the templates, determines if the cell converges at its maximum rate τ . τ is defined as the amount of time it takes for the state of a cell to change from its most positive state to its most negative state. The maximum convergence rate of a cell $C(i,j)$ is determined by C the capacitor, as defined in (1), and the equivalent resistance, R_{eq} , seen by C . If another cell $C(m,n)$ in the CNN has a variance in either C or R_{eq} the result will be a change in τ , defined as $\Delta\tau$.

The proper final state of a CNN can vary depending on $\Delta\tau$. If the output of a cell is used to determine the final state of a CNN, as is the case when $A \neq 0$, then it is possible that the cells can converge to a wrong value. The dependence of a CNN on $\Delta\tau$ is determined by the architecture of the array as well as the templates used to implement the given function.

If the CNN relies heavily on the B template, then the effect of $\Delta\tau$ is minimum. This is because the network is being driven by the constant value of u_{ki} representing the image which can offset any slowly converging cells.

The architecture used to implement a CNN can also increase the effects of $\Delta\tau$. If the slope of the nonlinear output limiting function $f_n(\cdot)$ is very steep, then the effects of $\Delta\tau$ are greatly increased. A very steep slope implies that the output, y , of a cell changes very rapidly when the value of the state, x , is close to the origin. Therefore, at some point in time, say $\frac{\tau}{2}$, the output of the cell changes very rapidly from -1 to $+1$ without the cell being fully converged. If a cell $C(i,j)$ has a convergence time of $\tau + \Delta\tau$, then using the same assumption, the output changes at time $\frac{\tau + \Delta\tau}{2}$. If cell $C(i,j)$ is surrounded by cells with the quicker convergence time of τ , then the outputs of the surrounding cells change states before cell $C(i,j)$.

If the output of cell $C(i,j)$ is dependent only on the output of the cells surrounding it, then the final state of cell $C(i,j)$ is effected by the improper data that it receives in the time period between $\frac{\tau}{2}$ and $\frac{\tau + \Delta\tau}{2}$. The following simulation results show this effect.

A 5x5 CNN was simulated using HSPICE. The template multipliers were voltage controlled current sources. The output function $f_n(\cdot)$ was implemented using a transistor level inverter. The slope of the output function is very steep as discussed above. The output, $y(t)$, transition from the negative rail to the positive rail takes place when the state, $x(t)$, is between -10mV and $+10\text{mV}$.

An edge detection template was used on the diamond image of Fig. 2a. The initial conditions placed on the cells are the values for the pixels of the image itself. A black pixel is represented by a normalized value of $+1$ volt and white pixel is represented by a normalized value of -1 volt. The simulation was ran with all cells having the same value of τ with $C=1.5\text{pF}$. The correct final state of the processed image is shown in Fig. 2b.

The τ of cell $C(3,3)$ was then altered by changing the value of the capacitor in the cell to $1.05C$. This had the effect of making $\Delta\tau$ equal to 5%. Fig. 2c shows that the slower convergence rate of cell $C(3,3)$ caused the CNN to converge to an improper final state.

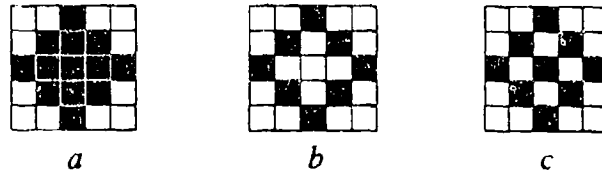


Fig. 2 Edge detection using initial conditions (a) input image (b) final results with $\Delta\tau = 0$ (c) final results with cell $C(3,3)$ having $\Delta\tau = 5\%$

Fig. 3 shows a comparison between the state of cell $C(3,3)$ and its surrounding cells. As the figure shows, at $1.7\mu s$ cell $C(3,3)$ has decayed to $.4V$ but the surrounding cells have already reached the transition point. Once the surrounding cells reach the transition point, their output changes to -1 . This change causes the surrounding cells to inject current into cell $C(3,3)$. The injected current causes the state of cell $C(3,3)$ to begin to rise. The graph shows the impact of this change on the cell.

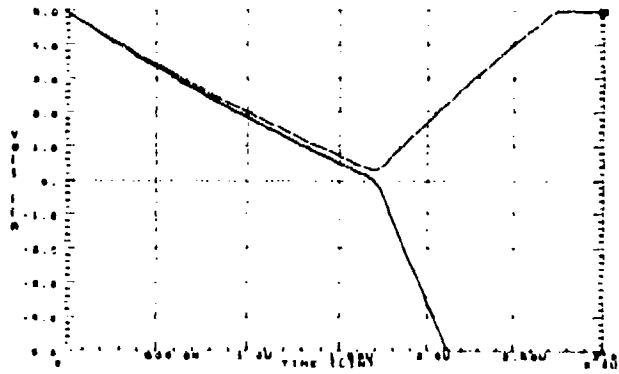


Fig. 3 State value comparison with cell $C(3,3)$ having a slower convergence time. The state of cell $C(3,3)$ is shown as the dashed line and the state of all surrounding cells is shown as the solid line.

4. Convergence Test Method

The test method presented next test a CNN for variances in convergence. The test is intended to be used in a production environment in conjunction with the functional test presented in the following sections. The test uses an edge detection type template to detect slowly converging cells. After the test, the slow converging cells appear black while the rest of the array turns white. The convergence test algorithm is presented next.

1. Let $A = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 4 & 1 \\ 0 & -1 & 0 \end{bmatrix}$ $B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ $i = -4.7$
2. Set the input image and border cells to black
 $\forall_{ij} u_{ij} = 1 \quad i = 0, 1, \dots, n+1; \quad j = 0, 1, \dots, m+1$
3. Set the initial conditions to black
 $\forall_{ij} x_{ij}(0) = -1 \quad i = 1, 2, \dots, n; \quad j = 1, 2, \dots, m$
4. Allow the CNN to converge
5. All cells, C_{ij} remaining at black are considered slow to converge
 $\forall_{ij} y_{ij} = -1 \rightarrow C_{ij} \quad i = 1, 2, \dots, n; \quad j = 1, 2, \dots, m$

The test works much in the same way as edge detection. The test image is a solid black box with the border cells forming the edge of the image. The initial conditions placed on the cells cause the dynamic convergence of the array to drive the cells to their final steady state. Cells which are slow to converge remain at black as discussed in the above example. The test can detect variances in $\Delta\tau$ as small as 5% as long as the cell does not have two or more slowly convergent orthogonal neighbors.

5. Simulated Convergence Test

A 5x5 CNN simulator implemented using HSPICE was used to confirm the above test procedure. All cells had a 1pF integrating capacitor except for cells $C(1,1)$, $C(2,3)$, $C(2,4)$, and cell $C(5,4)$ which had a capacitor value of 1.05pF. This resulted in a $\Delta\tau$ of 5%.



Fig. 4 Slow converging cells (a) original test image to be processed (b) final result showing cells $C(2,3)$, $C(2,4)$ and cell $C(5,4)$ are slow converging cells

As Fig. 4 shows, all slowly convergent cells remained black except cell $C(1,1)$. This is due to the fact that this cell had the constant driving force of the border cells to eventually force the cell to a white value. This is similar to the effect of having the B template present. Fig. 5 shows the state value of cell $C(1,1)$. The convergent rate of the cell is slowed when its neighbors outputs changes, but its direction is not reversed. Due to this phenomenon the corner cells of the array will always converge to the proper value.

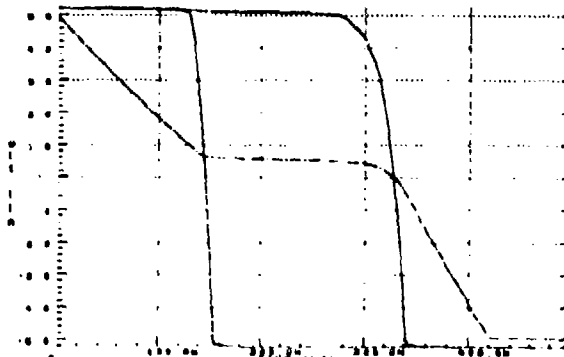


Fig. 5 State variable value of cell $C(1,1)$ showing the effect of the border cells on its convergence. The state of cell $C(1,1)$ is shown as —, and its output is shown as the solid line

6. Fault Models

A CNN processor has only two output states. Commonly, in image processing applications these states appear as white or black pixels. If a cell is unable to change from one state to the other, it is defined to be "stuck-at-white" or "stuck-at-black", depending on its current value. With the proposed test method it is possible to detect 100% of the stuck-at faults in the processor.

7. Functional Test Methods

The test procedure has two separate methods to detect faulty cells, a local method using the B template and a propagation method using the A template. The entire array can be tested using either of these methods regardless of its size. The advantage of the A template method is that it verifies that each cell is responding correctly to its neighbors output. The propagation test should be used if the processing array does not appear to be disseminating information throughout the network properly.

The local method uses the input image and the B template to predict the final output state, y_{ij} , of each cell in the array. The algorithm for the local test procedure is shown next.

1. Let $A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ $B = \begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ $I = 0$
2. Set the input image and border cells to white
 $\forall_{ij} u_{ij} = -1 \quad i = 0, 1, \dots, n+1; \quad j = 0, 1, \dots, m+1$
3. Set the initial conditions to white
 $\forall_{ij} x_{ij}(0) = -1 \quad i = 1, 2, \dots, n; \quad j = 1, 2, \dots, m$
4. Allow the CNN to converge
5. All cells, C_{ij} remaining at white are considered faulty
 $\forall_{ij} y_{ij} \equiv -1 \rightarrow C_{ij} \text{ is faulty} \quad i = 1, 2, \dots, n; \quad j = 1, 2, \dots, m$

The normalized value corresponding to a white input is -1 . The B template values are also -1 . The positive product of these two values results in current being injected into cell $C(i,j)$ from each of its eight neighbors. As the current is injected into the cell, the integrator voltage rises and the cell output reaches the normalized value of 1, which corresponds to black. If cell $C(i,j)$ fails to change under such overwhelming circumstances the conclusion that must be drawn is that the cell is "stuck-at-white". The result of this test does not depend on the output of any cell, therefore if cell $C(i,j)$ is "stuck-at-white" its output will not effect the output of its neighbors. The same algorithm can be applied to find "stuck-at-black" cells by changing all instances of white to black and -1 to 1 in steps 2 thru 5 of the local test algorithm.

The test of the CNN array using the A template uses the idea of propagation of information across the network. The propagation ability of CNNs has been described before [5]. Here we use the same concept although the templates are different since we only want propagation and not "full dragging" as described in [5]. When propagating information across the network, the effects of $\Delta\tau$ do not effect the final state of the CNN they only delay the final result due to the slower convergence time of any cells in the propagation path. In this case the input image does not matter and the border cells and initial conditions of the network are black. The A template shown in the algorithm below causes each cell, $C(i,j)$, to look at the cell behind it, $C(i-1,j)$, and change to the color of that cell. The algorithm for the propagation test method is outlined next.

The process starts at the left edge of the array and propagates across the network to the right side. Since the border cells, $C(i,0)$, are black, the predicted result should be an all black image. If a "stuck-at-white" fault is detected, then all properly functioning cells to the right of the stuck cell should also remain white. The faulty cell in effect cast its "shadow" across the array. The situation where two or more faulty cells lie on the same row can be detected by rotating the A template clockwise 45° and repeating the test. The template should be rotated in this manner 360° in order to assure complete coverage of the array. The "stuck-at-white" cells can be determined by performing the logical OR of the resulting eight output images. The same procedure can be used to detect "stuck-at-black" cells by changing all occurrences of white to black and -1 to 1 in steps 2 thru 8 in the propagation test algorithm and performing a logical AND in step 7.

8. Simulated Functional Test Results

Using a CNN simulation program the above tests were applied to a 5×5 CNN network. In both the local and propagation tests, cells $C(3,2)$ and $C(2,3)$ were intentionally forced into the "stuck-at-white" state. Fig. 6a shows the input image used for both tests. Fig. 6b shows the resulting final image after the simulation of the local test. It is clearly seen that all properly working cells have successfully implemented the B template and made the transition from white to black. Cell $C(2,2)$ was able to make the transition even though two of its

1. Let $A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ $B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ $1 = 0$ $k = 1$
2. Set the input image to white and the border cells to black

$$\begin{aligned} \forall_{ij} u_{ij} &= -1 & i &= 1, 2, \dots, n; & j &= 1, 2, \dots, m \\ \forall_{ij} u_{ij} &= 1 & i &= 0, n+1; & j &= 0, 1, 2, \dots, m+1 \\ \forall_{ij} u_{ij} &= 1 & i &= 0, 1, 2, \dots, n+1; & j &= 0, m+1 \end{aligned}$$
3. Set the initial conditions to black

$$\forall_{ij} x_{ij}(0) = 1 \quad i = 1, 2, \dots, n; \quad j = 1, 2, \dots, m$$
4. Allow the CNN to converge and save the results of rotation k

$$\forall_{ij} x_{ij}(t) = C_{ij}^{(k)} \quad i = 1, 2, \dots, n; \quad j = 1, 2, \dots, m$$
5. Add 1 to k and rotate A template clockwise 45°
6. If $k > 8$ continue to step 7, otherwise go to step 2
7. Perform the logical OR of the results

$$\bigvee_{k=1}^8 \forall_{ij} C_{ij}^{(k)} \rightarrow c_{ij} \quad i = 1, 2, \dots, n; \quad j = 1, 2, \dots, m$$
8. All cells, C_{ij} remaining at white are considered faulty

$$\forall_{ij} y_{ij} \equiv -1 \rightarrow C_{ij} \text{ is faulty} \quad i = 1, 2, \dots, n; \quad j = 1, 2, \dots, m$$

neighbors where faulty. This is because the output of each cell is due only to the input image and is independent of any cells output.

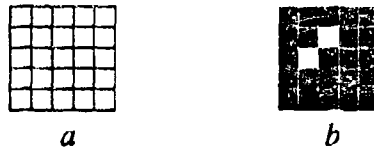


Fig. 6 Local test method (a) input image (b) final result

Fig. 7 shows the results of the simulation after the propagation test. Fig. 7a shows the "shadow" effect discussed earlier. It is safe to assume by viewing Fig. 7a that cells $C(3,2)$ and $C(2,3)$ are faulty. However, it is unclear if any of the remaining cells on rows 2 or 3 are "stuck-at-white" due to the "shadow" of the faulty cells. Fig. 7b shows the result after rotating the template 45° and the new direction of propagation. It is still unclear as to whether cell $C(3,4)$ is functioning properly. Fig. 7c confirms that cell $C(3,4)$ is functioning properly. For this example, ORing the three images of Figs. 7a-c is enough to show all faulty cells. In more complicated fault location patterns however, the template must be rotated completely to detect all possible faults. The results of the test are shown in Fig. 7i and it is clear that cells $C(3,2)$ and $C(2,3)$ are faulty.

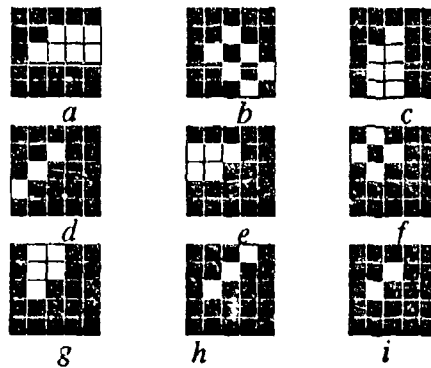


Fig. 7 Propagation test clockwise template rotation (a) 0° (b) 45° (c) 90° (d) 135° (e) 180° (f) 225° (g) 270° (h) 315° (i) final result

9. Conclusion

The variance in convergence rates of the cells of a CNN has been shown to have an impact on the final state of the Network. If cell $C(i,j)$ is slower to converge to its final state than the cells around it and if the architecture is sensitive to variances in τ , then cell $C(i,j)$ may fail to make the transition to its proper final value. A convergence test method to detect dependence on $\Delta\tau$ has been proposed. The test method can detect variances in convergence rates as small as 5%.

A testing method for CNNs has been presented which provides 100% fault detection with no additional hardware required. Only five input vectors are needed $x(0)$, u , A , B and I . The image vector u contains only two components, the color of the image and the color of the border cells. The initial conditions are always the same color for the entire array therefore the vector $x(0)$ needs to only represent the chosen color. The template vectors A and B always contain the nine values necessary to interact with the surrounding cells of $C(i,j)$. I , the independent current source vector is 0 in all cases. Since none of the input vectors have any dependence on the array size, any size array can be tested and the number and size of the input vectors will remain constant.

Both functional testing methods give 100% fault detection. The local testing method provides 100% fault isolation and the diagnosis is available by simply looking at the final image achieved after the CNN converges. There are some fault location configurations that could impede fault isolation using the propagation test, i.e. if the faults form a complete rectangle, the status of the cells inside the rectangle would be unknown due to the shadow effect. This fact lowers the fault isolation capabilities of the propagation method but, does not change the fault detection percentage. The diagnosis for the propagation test is available after the A template has been rotated 360° and the proper logic function has been performed on the results.

References

- [1] A. Rueda and J.L. Huertas, "Testability in Analogue Cellular Neural Networks", *Int. Journal of Circuit Theory and Applications*, 1992, 20, pp. 583-587.
- [2] W. Huang and F. Lombardi, "On an Improved Design Approach for C-Testable Orthogonal Iterative Arrays", *IEEE Transactions on Computer Aided Design*, May 1988, 7, no. 5, pp. 609-614.
- [3] L.O. Chua, and L. Yang, "Cellular Neural Networks: Theory", *IEEE Trans. Circuits and Systems*, 1988, CAS-35, pp. 1257-1272.
- [4] L.O. Chua and L. Yang, "Cellular Neural Networks: Applications", *IEEE Trans. Circuits and Systems*, 1988, CAS-35, pp. 1273-1290.
- [5] T. Matsumoto, L.O. Chua and H. Suzuki, "CNN Cloning Template: Connected Component Detector", *IEEE Trans. Circuits and Systems*, May 1990, 37, no. 5, pp. 633-635.